L Number	Hits	Search Text	DB	Time stamp
-	5147	access$3 adj2 list$1	USPAT; US-PGPUB	2004/03/10 13:17
-	160	router$1 adj2 name$1	USPAT; US-PGPUB	2004/03/10 13:18
-	0	((access$3 adj2 list$1) and (router$1 adj2 name$1)) and @ad<19960621	USPAT; US-PGPUB	2004/03/10 13:22
-	19	(access$3 adj2 list$1) and (router$1 adj2 name$1)	USPAT; US-PGPUB	2004/03/10 13:19
-	1088	(access$3 adj2 list$1) and @ad<19960621	USPAT; US-PGPUB	2004/03/10 13:22
-	5589	filter$3 near5 address$3	USPAT; US-PGPUB	2004/03/10 13:23
-	18	((access$3 adj2 list$1) and @ad<19960621) and (filter$3 near5 address$3)	USPAT; US-PGPUB	2004/03/10 13:23

US-PAT-NO: 5509123

DOCUMENT-IDENTIFIER: US 5509123 A

TITLE: Distributed autonomous object architectures for network layer routing

---------- KWIC ---------

Application Filing Date - AD (1):
19940322

Detailed Description Text - DETX (12):
B.3 **Access List**

Detailed Description Text - DETX (28):
Returning to FIG. 3B, the forwarding information base 233' is connected to each of a plurality of "protocol" forwarding engines 234' which are further illustrated in FIG. 3E. The protocol forwarding engine object 234' includes a forward and service object 239' connected to each of an **access list** object 240', a next-hop cache object 241', and a framing object 242'. The framing object 242' is connected to each of address cache object 243' and network interface object 237'. The next-hop cache object 241' is connected to the network forwarding table object 233', or FIB in FIG. 3B. Returning to FIG. 3B, each network interface object 237' is connected below to a network media device driver 238'.

Detailed Description Text - DETX (38):
In this prior model, each protocol always gets the packet and then decides if it was appropriate for the interface. It is a centralized model with the protocol layer being the funnel for all packets entering and exiting the system regardless of the interface the packet came in on. Also, because it is centralized, each layer must have knowledge about every specific interface. For example, all configured interface information such as MTU size, forwarding enabled/disabled state, configured network addresses and masks, data-link framing options, filter **access lists,** etc, must be accessed by each layer as it processes/forwards the packet. This model puts overhead into each layer and is very limiting in supporting new interfaces, media, and protocols, as each layer must be modified. An example of overhead is that if a packet is received for a protocol that is not enabled, it is not dropped until it has been passed up to the correct protocol layer.

Detailed Description Text - DETX (82):
As part of forwarding packets, the IP forwarding engine methods (1) validate packet **addresses, (2) filter** against an **access list,** and (3) retrieve the next

hop from the FIB. These procedures are inherently slow, so the results of these procedures once obtained, such as address validity, are cached and corresponding procedures are provided in IPACache to lookup the same results quickly.

Detailed Description Text - DETX (85):
access control list filtering

Detailed Description Text - DETX (87):
Each of these procedures is passed the source and destination addresses from a packet, hashes them and looks up entries linked in the "bucket" for that hash code. It checks each linked entry to see if it matches exactly both the source and the destination. If it finds a match it returns the entry data for that function. For the Martian lookup the address validity, yes or no, is returned. For access control lookup (see **access list** object 240' in FIG. 3E) an additional protocol and port parameter must be matched and permission, permit or deny, is returned. For next hop a quality of service parameter must be matched and the next hop is returned.

Detailed Description Text - DETX (88):
These lookup procedures are called in the context of an interrupt service routine attempting to forward the packet; they are coded to be fast. However, the cache may be temporarily inactive--say it is being flushed in a thread context due to an external management event, such as the deletion of an **access list** entry. In this case, the cache lookup routine simply falls back on the original, slow procedure which made the decision when the entry was cached, effectively bypassing the cache.

Detailed Description Text - DETX (96):
B.3 **ACCESS LIST**

Detailed Description Text - DETX (97):
In general, routing applications allow network management to **filter packets based on destination address,** or on the combination of destination and source addresses. It is desirable that each interface of the router be able to maintain a separate set of filter instances--an **access list**. Most vendors use a linear mechanism and since the list must be checked for each packet being forwarded, throughput slows down linearly as the list gets larger.

Detailed Description Text - DETX (99):
To provide an object-oriented, powerful and very efficient access control mechanism, a base class FAC (Forwarding Access) was invented. For efficiency, FAC keeps **access list** entries as nodes in an AVL tree. A tree does not have a predefined size and may grow freely. Management routines are provided to allow network management to set entries and retrieve them in serial order. These routines are supported by iterator classes written to walk the AVL trees.

Detailed Description Text - DETX (100):
Efficiency is further maximized because within each **access list,** valid entries are also linked across the tree in their sequence order for fast scan during filtering. Each interface may associate with one and only one **access list** as identified by an ID in the entry. This association is done in the forwarding part of a protocol's MIB via an ID leaf and a control leaf to enable or disable filtering.

Detailed Description Text - DETX (102):
FAC is a base class for all protocols' access control including IP, IPX, DECnet, Appletalk and for filtering routing protocols such as RIP. Each individual protocol derives its **access list** class from FAC. In the protocol-specific MIBs, access control consists of a table of entries. An entry begins with four common leaves managed in FAC: ID, Sequence, Matches and Permission. The first two index the entry instance and are unique to that entry.

Detailed Description Text - DETX (104):
Sequence: The sequence number keys the order of entries in a given **access list**. When filtering a packet, the first matching entry exits the filter check and a packet may match multiple entries, so order is important.

Detailed Description Text - DETX (113):
This allows the same list to be associated with multiple interfaces (but not necessarily all interfaces) so that fewer **access list** entries need be created.

Detailed Description Text - DETX (115):
Although addressing is the protocol-specific part of the **access list** entry, all protocol FAC derived classes support some special case address values which stand for a range of addresses. For example, in IP an address is paired with a mask and 0's in the mask are wild cards matching anything in the corresponding part of the address. Thus an address paired with a mask of all 0's matches everything. This is powerful--to filter out all packets from any source destined to a server, set the **access list** entry with the server's destination address and mask of all 1's, but use a source address and mask of 0's.

Detailed Description Text - DETX (200):
Access Control Group--contains the managed objects that pertain to establishing **Access Control Lists** for the network protocol's traffic.

US-PAT-NO: 5845091

DOCUMENT-IDENTIFIER: US 5845091 A

TITLE: Forwarding of internetwork packets to a destination network via a selected one of a plurality of paths

---------- KWIC ---------

Application Filing Date - AD (1):
19960215

Detailed Description Text - DETX (2):
As described herein, multiple paths are provided between a source network and a destination network, and a router is provided with a traffic filter that allows the user to select the path that traffic to the destination network will follow. According to one embodiment, **filtering is performed solely on the basis of the destination address** of a packet. If a packet is destined for the destination network, a forwarding list is consulted, and the packet is forwarded as indicated by the forwarding list. According to an alternative embodiment, the source network is subdivided into a multiplicity of subnetworks, and each subnetwork is provided with its own forwarding list. Thus, **filtering is performed on the basis of the source address** and the destination address of the packet. The filtering table is selected by the user to specify a default path, and the remaining paths are ordered at the preference of the user.

Detailed Description Text - DETX (15):
Thus, FIG. 9 shows the application of the criteria used by the filter to process a packet. At process block 905, router 801 receives a packet, and router 801 performs the datalink layer processing at process block 910. The criteria applied to the data packet is basically shown at process block 915, wherein router 801 determines whether the destination IP address is equal to the destination IP address of network B. If the received packet includes a destination IP **address that is not that of network B, the process of the filter** ends at process block 920, otherwise the filtering function is performed at step A, which is continued in FIG. 10.

Detailed Description Text - DETX (16):
According to the present embodiment, a forwarding list associated with network B is maintained in router 801. At process block 1005, the routing engine **accesses the forwarding list** to determine which path the received packet is to be transmitted over. At process block 1010, the routing engine determines whether the primary path indicated by the list pointer is currently valid. If the primary path is currently valid, the packet is sent via the primary path at process block 1015. If the primary path is not valid, it is

determined whether all entries of the forwarding list have been checked for validity, and if they have not, the list pointer is incremented at process block 1020 and is determined whether process steps 1005 and 1010 are repeated. If all entries of the forwarding list have been checked for validity and found to be invalid, the routing engine of router 801 attempts to reestablish the path indicated by each entry of the forwarding list. This may be done, for example, by transmitting an ARP request to each of the routers indicated by the forwarding list.

Claims Text - CLTX (30):

a routing engine coupled to the ports for processing packets received from the plurality of ports, the routing engine including a plurality of separate forwarding lists corresponding respectively to each of the plurality of subnetworks, each forwarding list including a plurality of entries specifying a possible path to a destination and a filter that, when a packet destined for a destination is received, selects a forwarding list based on a source **address of the packet, said filter** forwards the packet to a destination via a default path if a first entry indicates a valid path, said filter forwards the packet via a second path if the default path indicates an invalid path, said filter automatically resets said first entry as said default path for a second packet.

US-PAT-NO: 5790554

DOCUMENT-IDENTIFIER: US 5790554 A

TITLE: Method and apparatus for processing data packets in a network

---------- KWIC ---------

Application Filing Date - AD (1):
19951004

Brief Summary Text - BSTX (20):
A technique that has been employed by prior art network devices such as a LAN switch involves **access lists,** or filters, that allow the network administrator to control the forwarding of packets from a network device based upon the contents of the data packet. Such **access lists** allow a user to define a value within a specific field of a data packet. For example, to filter on an Internet protocol (IP) data packet with an IP address of 129.1.1.1, a user may configure and then apply to a particular port an **access list** that forwards or drops data packets having a value of 129.1.1.1 in the IP header of the data packet.

Brief Summary Text - BSTX (22):
Prior art filtering mechanisms allow for the application of multiple filters to the same data packet; however, the filters are applied in sequential order--no skipping to other filters is allowed. As soon as a match is found, no further filters are considered and the packet is processed according to the filter for which a match occurred. The only processing provided is to either permit the packet to be forwarded or drop the packet. There is no mechanism by which the data packet may be redirected to a port of the network device other than the normal destination port to which the packet is forwarded in the absence of an **access list** or filter, nor is a packet redirected to multiple destination ports.

Brief Summary Text - BSTX (23):
There are a number of disadvantages to the above approach for controlling the flow of data packets in a network device. A network administrator must specify a well known field based on an **access list** type, i.e., the manager is not allowed to specify an arbitrary offset within the data packet at which to compare the contents of the data packet to a value specified by the filter. Moreover, a filter cannot jump to another filter, rather, filters are applied according to the order in which they are configured in the network device. Furthermore, prior art filtering systems do not allow forwarding of a data packet to an alternative port or an additional port. The packets may only be forwarded to the normal destination port or dropped. Finally, filters

heretofore have only allowed the logical operators equal and not equal in determining whether a value specified by the filter matches or fails to match the contents of a data packet at the location in the packet specified by the filter. The additional logical operators of less than, less than or equal to, greater than, and greater than or equal to, have not been permissible.

Detailed Description Text - DETX (21):
In one embodiment, each of the fields in the RIF are assigned a number. If the routing information indicator (RII) bit in the MAC source **address field is not set, the RIF type filter** will fail. The number assigned to each field in the RIF may be as follows:

Other Reference Publication - OREF (1):
"Create **Access Lists,**" pp. 17-18, Chapter 5--Managing the System, Router Products Configuration Guide, Cisco Systems.

US-PAT-NO: 5751971

DOCUMENT-IDENTIFIER: US 5751971 A
See image for Certificate of Correction

TITLE: Internet protocol (IP) work group routing

---------- KWIC ---------

Application Filing Date - AD (1):
19950712

Brief Summary Text - BSTX (12):
With ever increasing numbers of subnets, it would be desirable if further methods were available to conserve on subnet addresses. One potential method for doing this would be to put a bridge on a single router interface to bridge multiple LAN segments; however, this involves the added cost of a bridge and loses the protection of router "fire walls", which administrators set to **filter out packets based on destination addresses**. Another potential method would be to increase the granularity of subnets by taking more bits from the host portion of the IP address for the subnet mask; however, this approach is very difficult for the network administrator to maintain as the network configuration evolves. Thus, neither of these potential methods offers a satisfactory solution.

Detailed Description Text - DETX (73):
More specifically, the address range must lie within the subnet defined for a given work group and thus the entry acquires the security level of that work group. If security is violated, packets to and from a given host IP **address will be filtered** out by the router. The source and destination IP packet addresses are checked against ranges in the Range table during packet forwarding and must match as follows:

Detailed Description Text - DETX (100):
The operation of the forwarding engine will now be described with regard to FIG. 7. When an IP packet arrives, physically addressed to router interface-1, it is delivered to that FAS's service routine. The service routine validates the IP header and IP **addresses, filters** against any **access list,** and then looks up the destination address in the forwarding information base (FIB) to find a route. A route gives: (a) the outgoing interface; (b) the outgoing FAS, and if the destination is not directly connected to that interface; (c) the next hop router. If there is a valid route the service routine passes the packet to the forward routine of the outgoing FAS.

Detailed Description Text - DETX (101):

The forward routine of the outgoing FAS filters against its **access list** if any, and then tries to resolve the destination IP address or next hop to a physical address suitable for framing by looking in the ARP (address resolution protocol) cache associated with that interface. If the address is resolved the packet is transmitted to that physical address. Otherwise, the packet is deferred on an ARP entry queue and ARP tries to resolve the address through protocol request. If resolved, the deferred packet is dequeued and transmitted by the FAS.

Detailed Description Text - DETX (102):
A cache of packet forwarding history is kept by each FAS, keyed by destination and source IP addresses. **Address validation, access control filtering** and look-up of next hop check the cache first, and if an entry is found there, the method is quick. If at any stage an error occurs in forwarding, the packet is dropped and an ICMP control message is sent back to the source.

Detailed Description Text - DETX (135):
do **Access List** Control filter check;

Detailed Description Text - DETX (151):
do **Access List** Control filter check;

US-PAT-NO: 5699513

DOCUMENT-IDENTIFIER: US 5699513 A

TITLE: Method for secure network access via message intercept

---------- KWIC ---------

Application Filing Date - AD (1):
19950331

Drawing Description Text - DRTX (4):
FIG. 2 shows an exemplary **access control list** which controls the operation of a filter;

Detailed Description Text - DETX (5):
FIG. 2 shows an exemplary **access control list** (ACL) 34 which controls the operation of filter 16. ACL 34 represents a table stored in memory (not shown) of the node that implements filter 16. Generally, ACL 34 associates various items which are typically included with message control data in IP and other packet headers. These message or packet control data items are associated with actions that filter 16 may take.

Detailed Description Text - DETX (7):
As shown at an entry 48, when a packer's source and destination addresses indicate an entities in outside network 12, filter 16 may be programmed to block the packet so that it cannot enter inside network 14. An entry 50 indicates that packets having a source address associated with inside network 14 and a destination address associated with outside network 12 may pass through filter 16. An entry 52 indicates that connection request packets with a specified outside source address and an inside network 14 address are forwarded to logging port 32 (see FIG. 1). In accordance with TCP/IP terminology, a connection request message or packet is called a sync packet. An entry 54 indicates that other types of packets with a specified outside source address and an inside network 14 **address are passed through filter** 16.